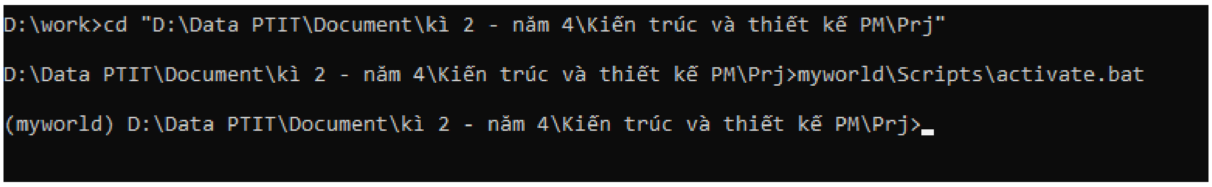
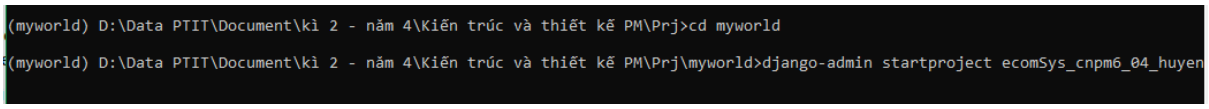
1. Trỏ vào và activate môi trường ảo

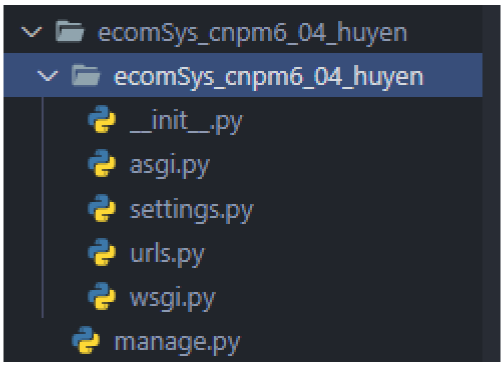


```
D:\work>cd "D:\Data PTIT\Document\kì 2 - năm 4\Kiến trúc và thiết kế PM\Prj"

D:\Data PTIT\Document\kì 2 - năm 4\Kiến trúc và thiết kế PM\Prj>myworld\Scripts\activate.bat

(myworld) D:\Data PTIT\Document\kì 2 - năm 4\Kiến trúc và thiết kế PM\Prj>
```

2. Tạo project:

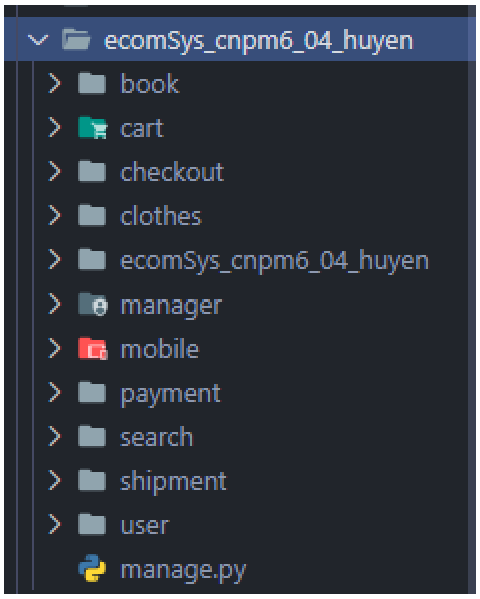


```
(myworld) D:\Data PTIT\Document\kì 2 - năm 4\Kiến trúc và thiết kế PM\Prj>cd myworld

(myworld) D:\Data PTIT\Document\kì 2 - năm 4\Kiến trúc và thiết kế PM\Prj\myworld>django-admin startproject ecomSys_cnpm6_04_huyen
```



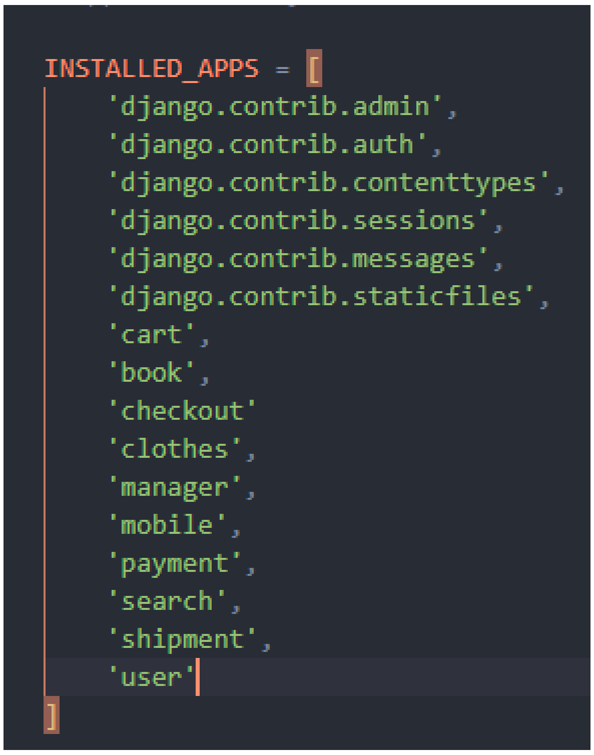
```
ecomSys_cnpm6_04_huyen
  ecomSys_cnpm6_04_huyen
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

3. Tạo 10 apps

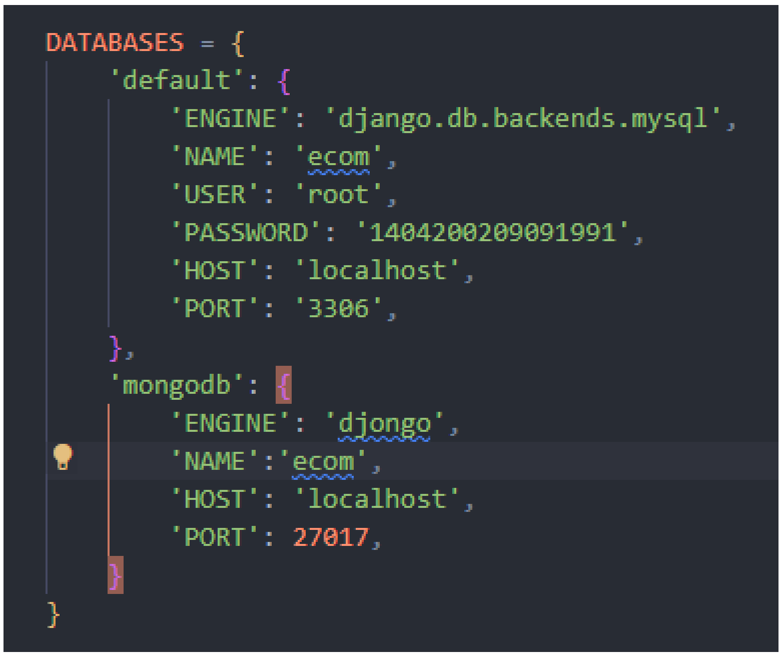


```
ecomSys_cnpm6_04_huyen
  book
  cart
  checkout
  clothes
  ecomSys_cnpm6_04_huyen
  manager
  mobile
  payment
  search
  shipment
  user
  manage.py
```

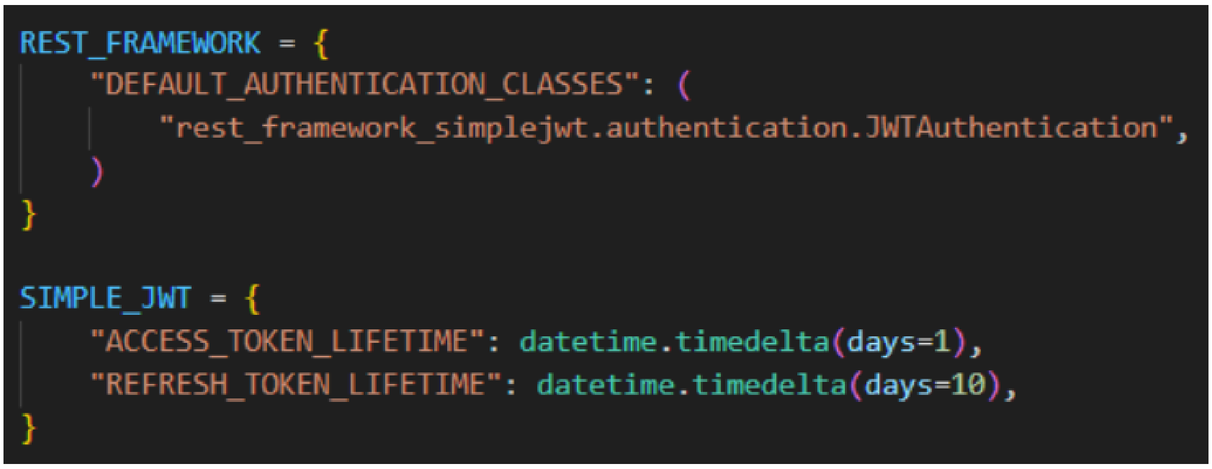
4. File settings.py



```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'cart',
    'book',
    'checkout'
    'clothes',
    'manager',
    'mobile',
    'payment',
    'search',
    'shipment',
    'user'
]
```



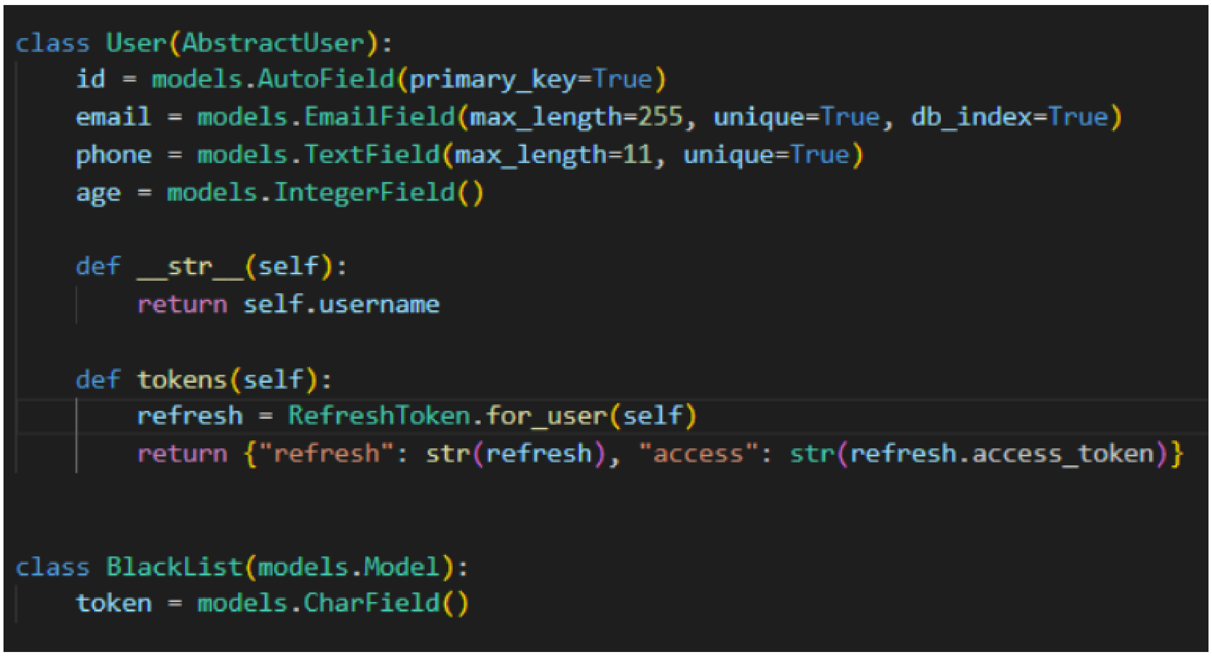
```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'ecom',
        'USER': 'root',
        'PASSWORD': '14042002090991991',
        'HOST': 'localhost',
        'PORT': '3306',
    },
    'mongodb': {
        'ENGINE': 'djongo',
        'NAME':'ecom',
        'HOST': 'localhost',
        'PORT': 27017,
    }
}
```



```python
REST_FRAMEWORK = {
    "DEFAULT_AUTHENTICATION_CLASSES": (
        "rest_framework_simplejwt.authentication.JWTAuthentication",
    )
}

SIMPLE_JWT = {
    "ACCESS_TOKEN_LIFETIME": datetime.timedelta(days=1),
    "REFRESH_TOKEN_LIFETIME": datetime.timedelta(days=10),
}
```

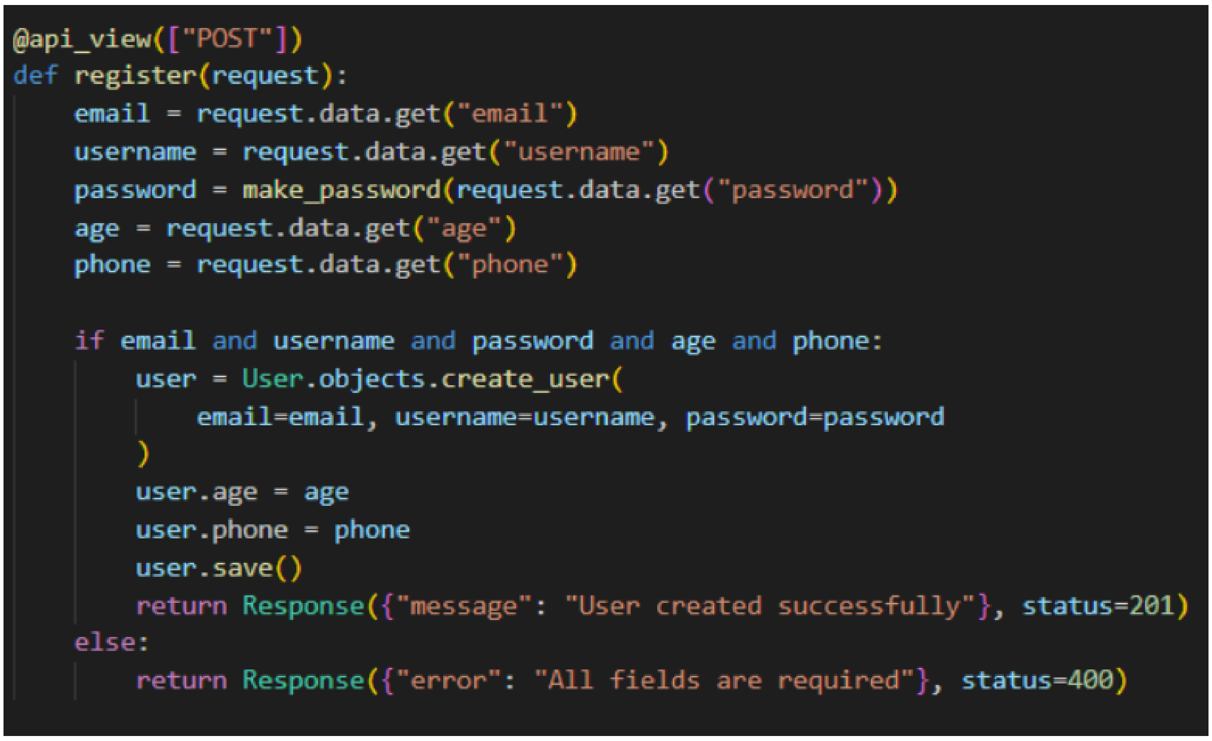
5. App User

   Model:

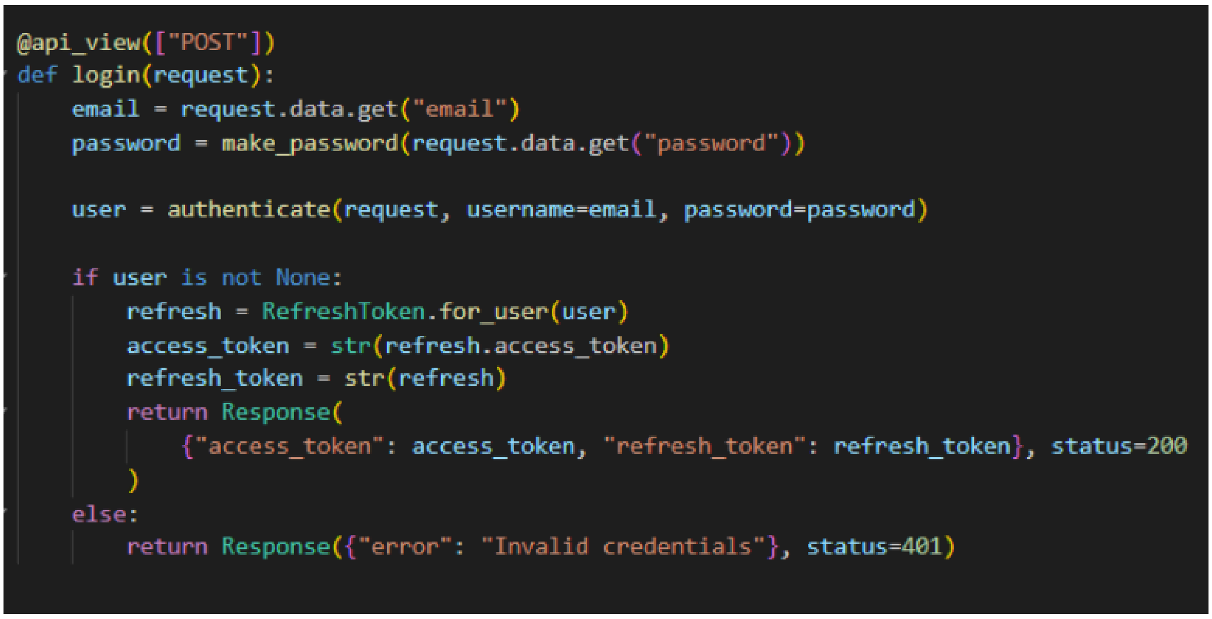


```python
class User(AbstractUser):
    id = models.AutoField(primary_key=True)
    email = models.EmailField(max_length=255, unique=True, db_index=True)
    phone = models.TextField(max_length=11, unique=True)
    age = models.IntegerField()

    def __str__(self):
        return self.username

    def tokens(self):
        refresh = RefreshToken.for_user(self)
        return {"refresh": str(refresh), "access": str(refresh.access_token)}


class BlackList(models.Model):
    token = models.CharField()
```

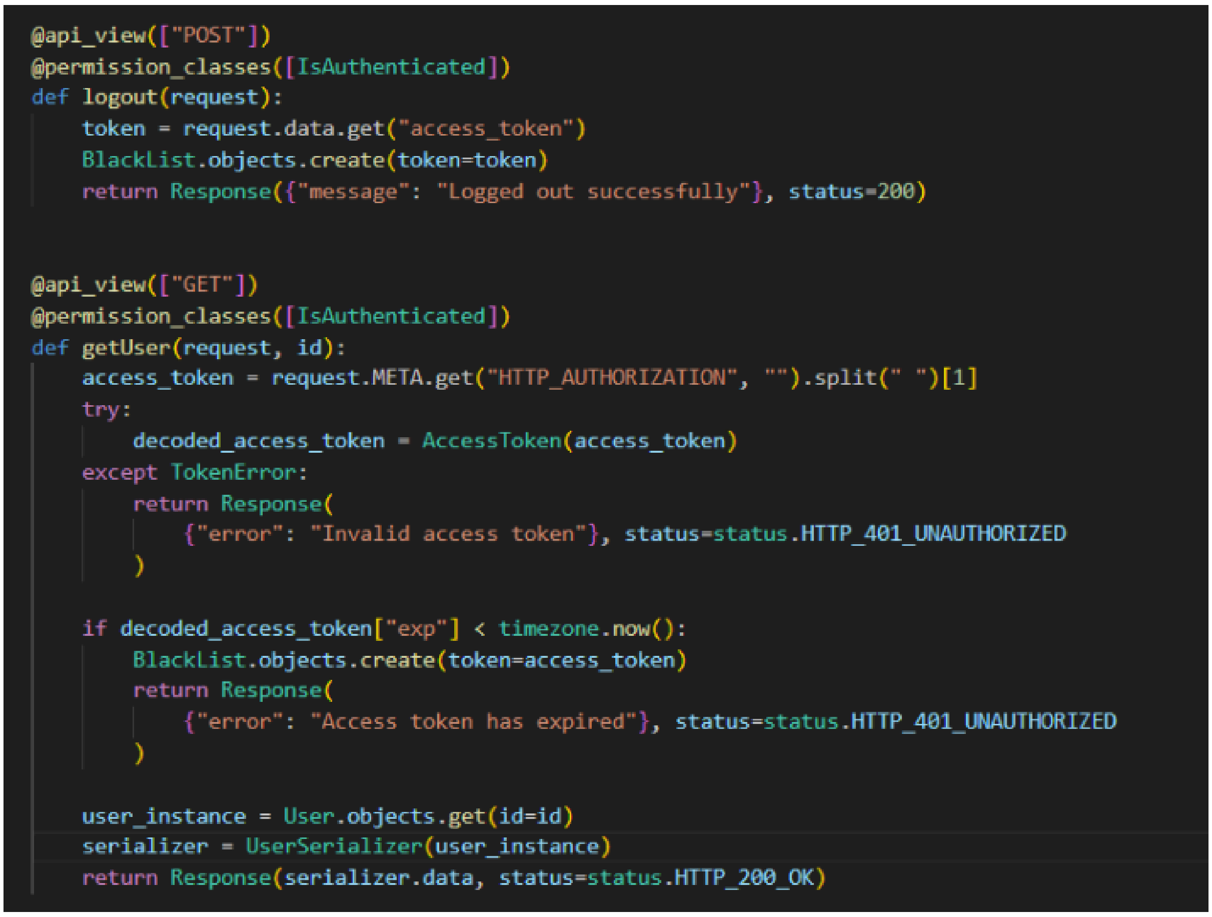
   View:



```python
@api_view(["POST"])
def register(request):
    email = request.data.get("email")
    username = request.data.get("username")
    password = make_password(request.data.get("password"))
    age = request.data.get("age")
    phone = request.data.get("phone")

    if email and username and password and age and phone:
        user = User.objects.create_user(
            email=email, username=username, password=password
        )
        user.age = age
        user.phone = phone
        user.save()
        return Response({"message": "User created successfully"}, status=201)
    else:
        return Response({"error": "All fields are required"}, status=400)
```

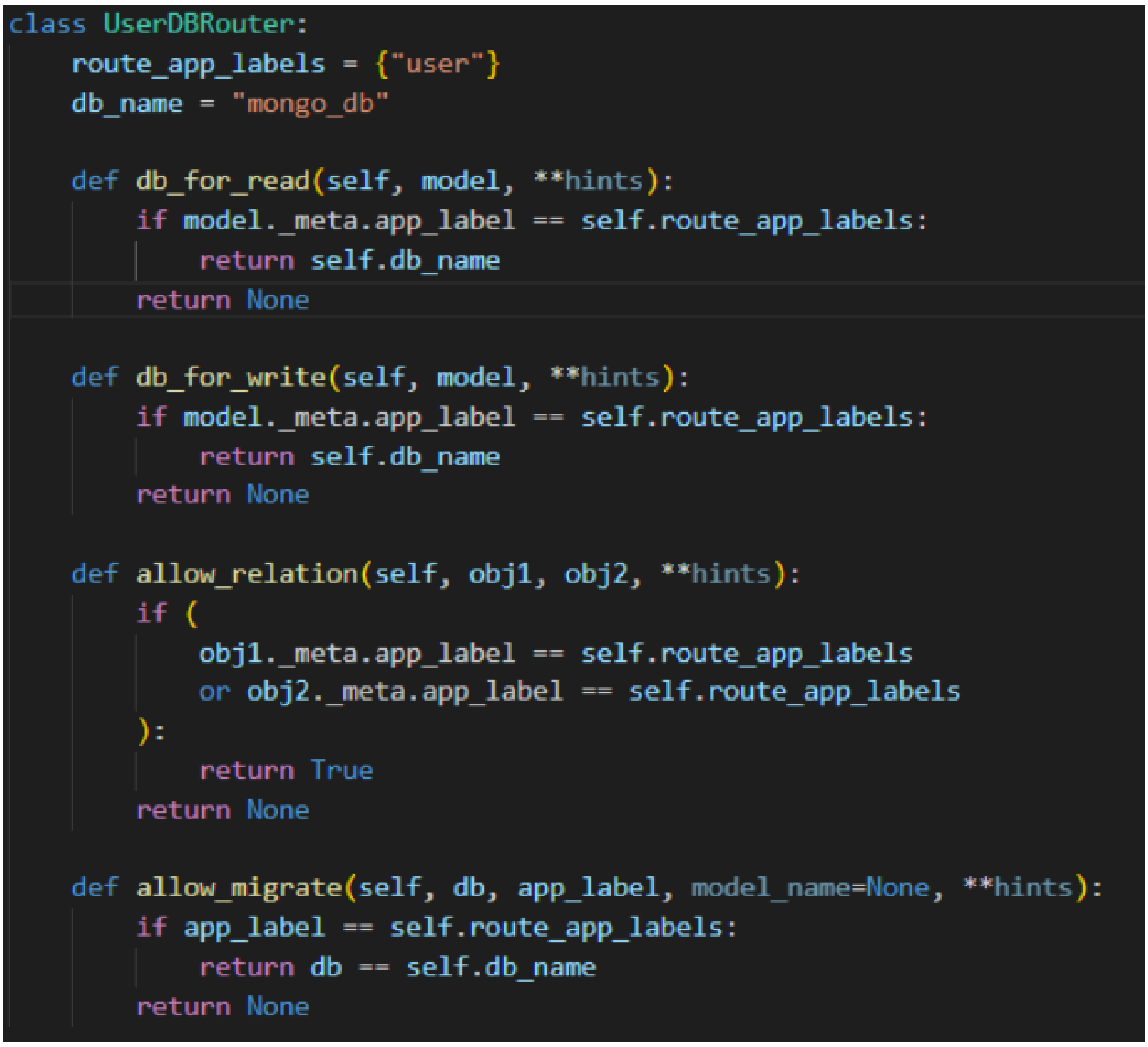


```python
@api_view(["POST"])
def login(request):
    email = request.data.get("email")
    password = make_password(request.data.get("password"))

    user = authenticate(request, username=email, password=password)

    if user is not None:
        refresh = RefreshToken.for_user(user)
        access_token = str(refresh.access_token)
        refresh_token = str(refresh)
        return Response(
            {"access_token": access_token, "refresh_token": refresh_token}, status=200
        )
    else:
        return Response({"error": "Invalid credentials"}, status=401)
```



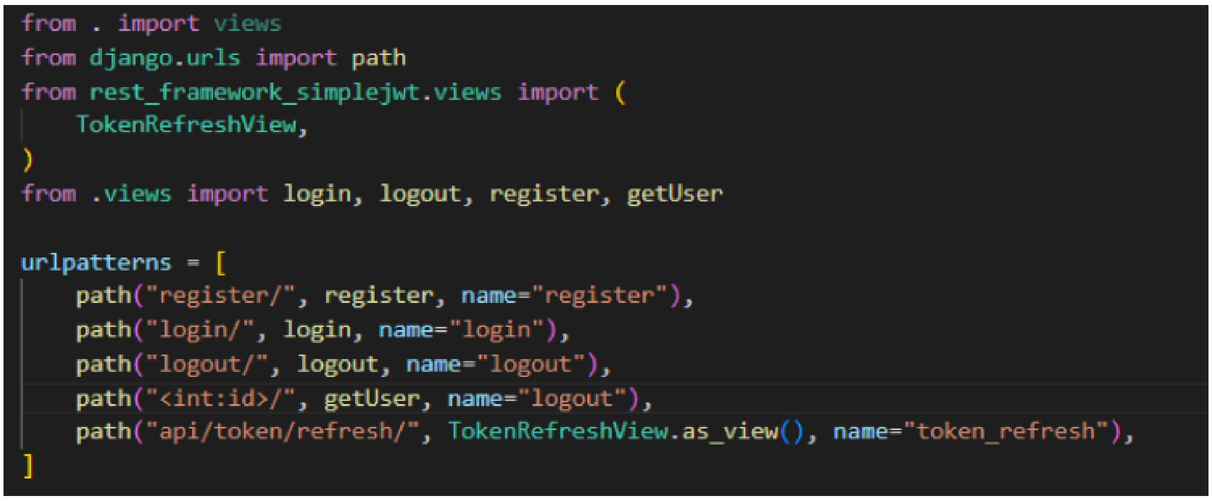
```python
@api_view(["POST"])
@permission_classes([IsAuthenticated])
def logout(request):
    token = request.data.get("access_token")
    BlackList.objects.create(token=token)
    return Response({"message": "Logged out successfully"}, status=200)


@api_view(["GET"])
@permission_classes([IsAuthenticated])
def getUser(request, id):
    access_token = request.META.get("HTTP_AUTHORIZATION", "").split(" ")[1]
    try:
        decoded_access_token = AccessToken(access_token)
    except TokenError:
        return Response(
            {"error": "Invalid access token"}, status=status.HTTP_401_UNAUTHORIZED
        )

    if decoded_access_token["exp"] < timezone.now():
        BlackList.objects.create(token=access_token)
        return Response(
            {"error": "Access token has expired"}, status=status.HTTP_401_UNAUTHORIZED
        )

    user_instance = User.objects.get(id=id)
    serializer = UserSerializer(user_instance)
    return Response(serializer.data, status=status.HTTP_200_OK)
```

Router:



```python
class UserDBRouter:
    route_app_labels = {"user"}
    db_name = "mongo_db"

    def db_for_read(self, model, **hints):
        if model._meta.app_label == self.route_app_labels:
            return self.db_name
        return None

    def db_for_write(self, model, **hints):
        if model._meta.app_label == self.route_app_labels:
            return self.db_name
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if (
            obj1._meta.app_label == self.route_app_labels
            or obj2._meta.app_label == self.route_app_labels
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label == self.route_app_labels:
            return db == self.db_name
        return None
```

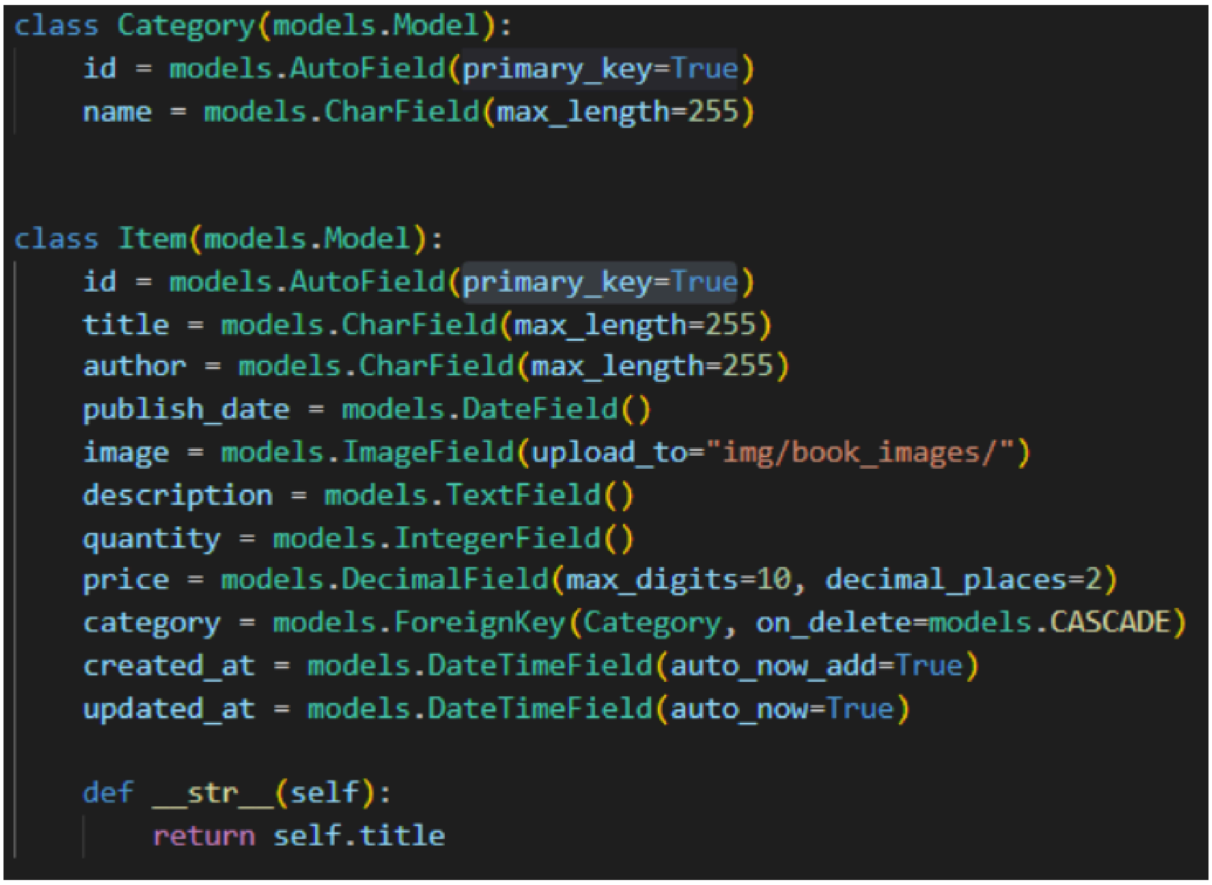
Url:



```python
from . import views
from django.urls import path
from rest_framework_simplejwt.views import (
    TokenRefreshView,
)
from .views import login, logout, register, getUser

urlpatterns = [
    path("register/", register, name="register"),
    path("login/", login, name="login"),
    path("logout/", logout, name="logout"),
    path("<int:id>/", getUser, name="logout"),
    path("api/token/refresh/", TokenRefreshView.as_view(), name="token_refresh"),
]
```

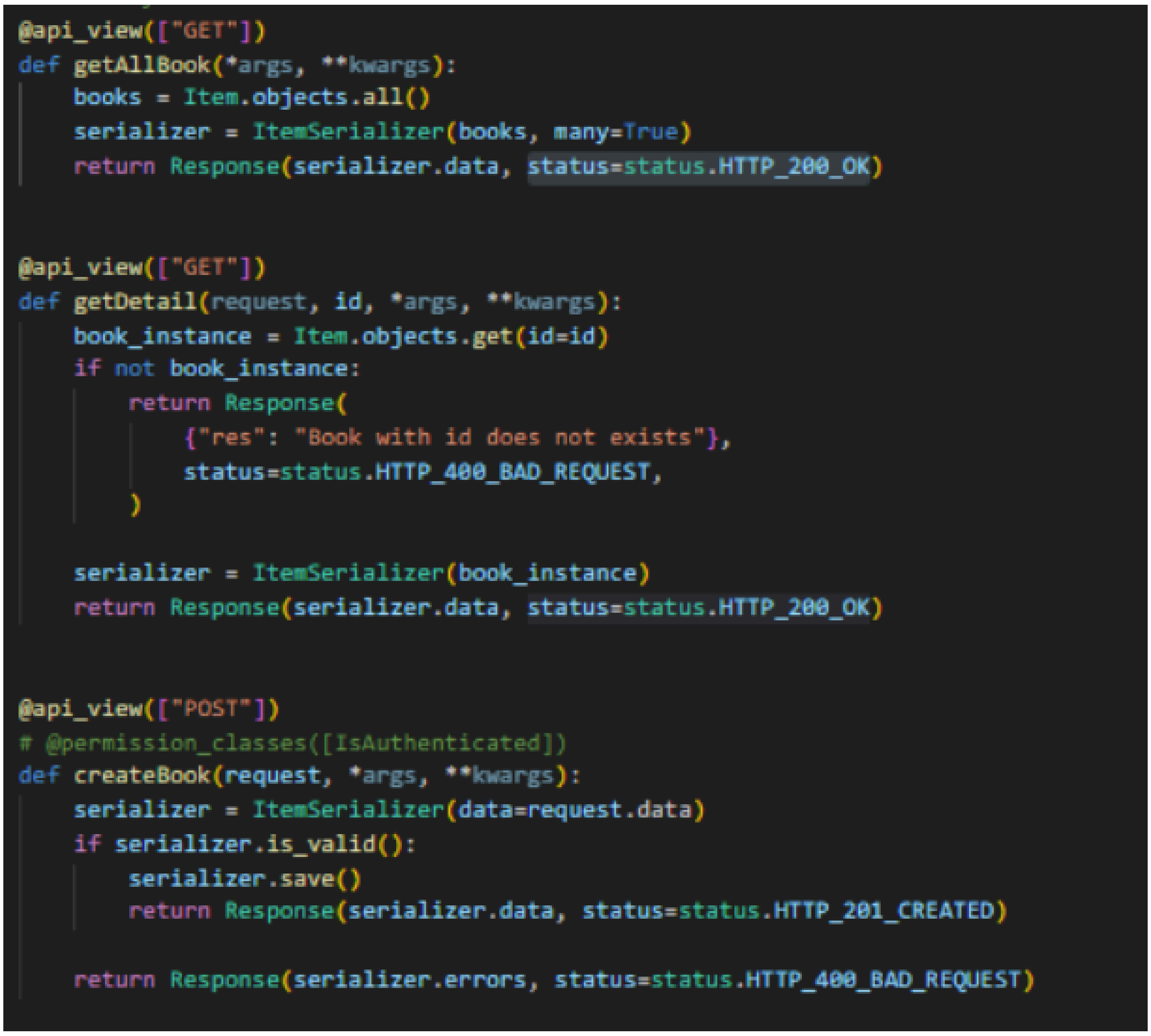
6. App Book

Model:

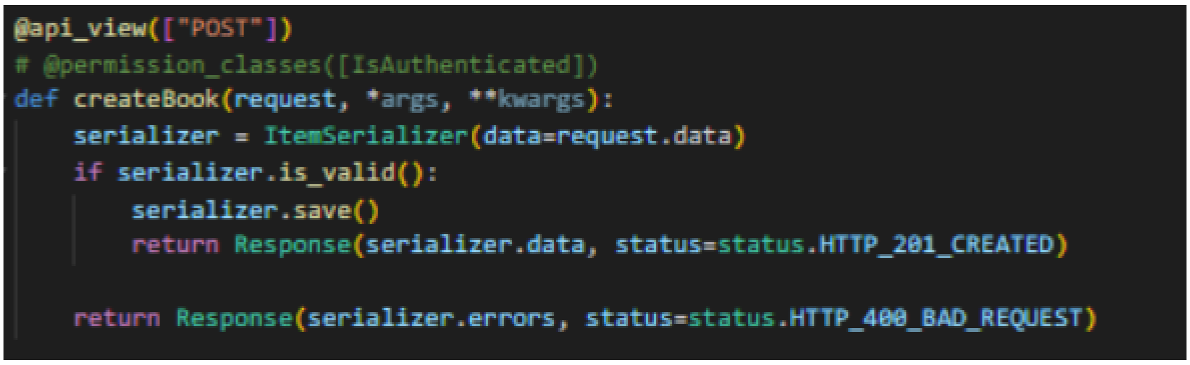


```python
class Category(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=255)


class Item(models.Model):
    id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=255)
    author = models.CharField(max_length=255)
    publish_date = models.DateField()
    image = models.ImageField(upload_to="img/book_images/")
    description = models.TextField()
    quantity = models.IntegerField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.title
```

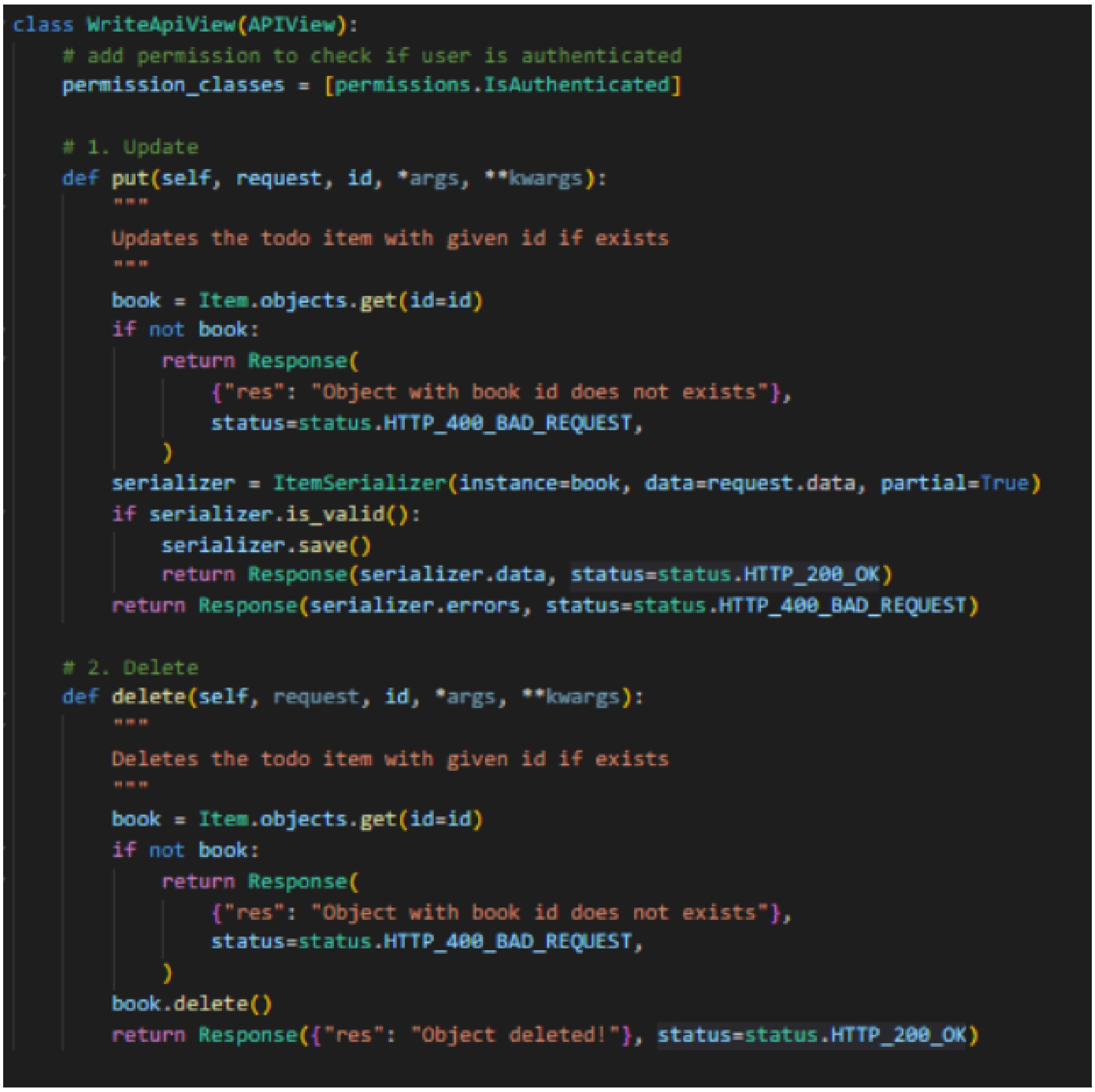
View:



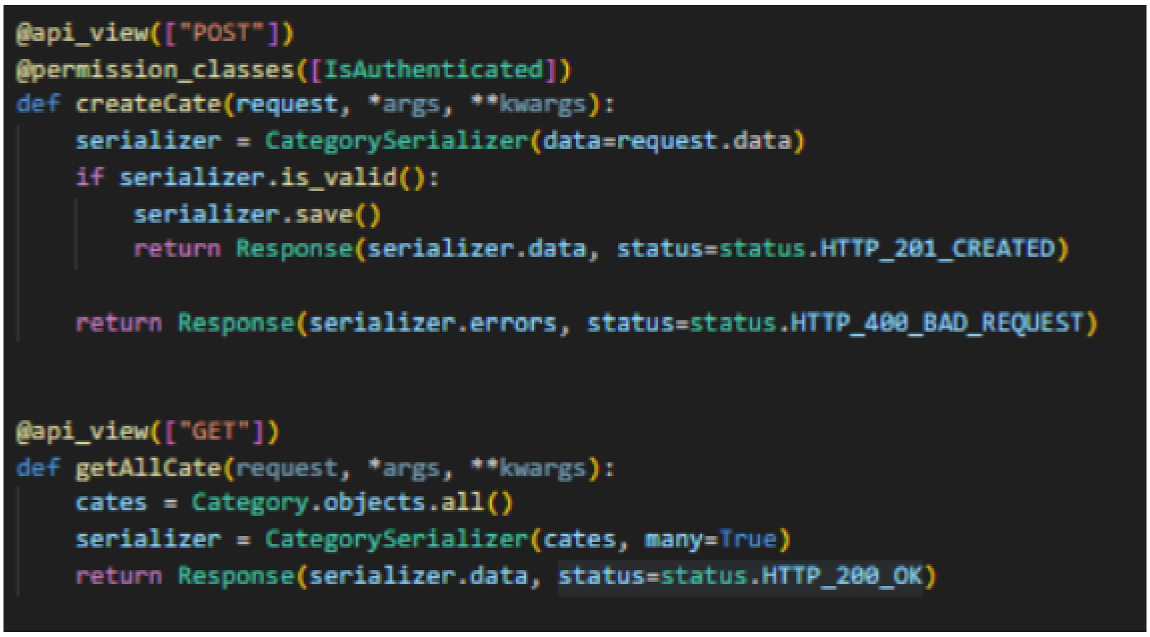
```python
@api_view(["GET"])
def getAllBook(*args, **kwargs):
    books = Item.objects.all()
    serializer = ItemSerializer(books, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["GET"])
def getDetail(request, id, *args, **kwargs):
    book_instance = Item.objects.get(id=id)
    if not book_instance:
        return Response(
            {"res": "Book with id does not exists"},
            status=status.HTTP_400_BAD_REQUEST,
        )

    serializer = ItemSerializer(book_instance)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createBook(request, *args, **kwargs):
    serializer = ItemSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```



```python
@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createBook(request, *args, **kwargs):
    serializer = ItemSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

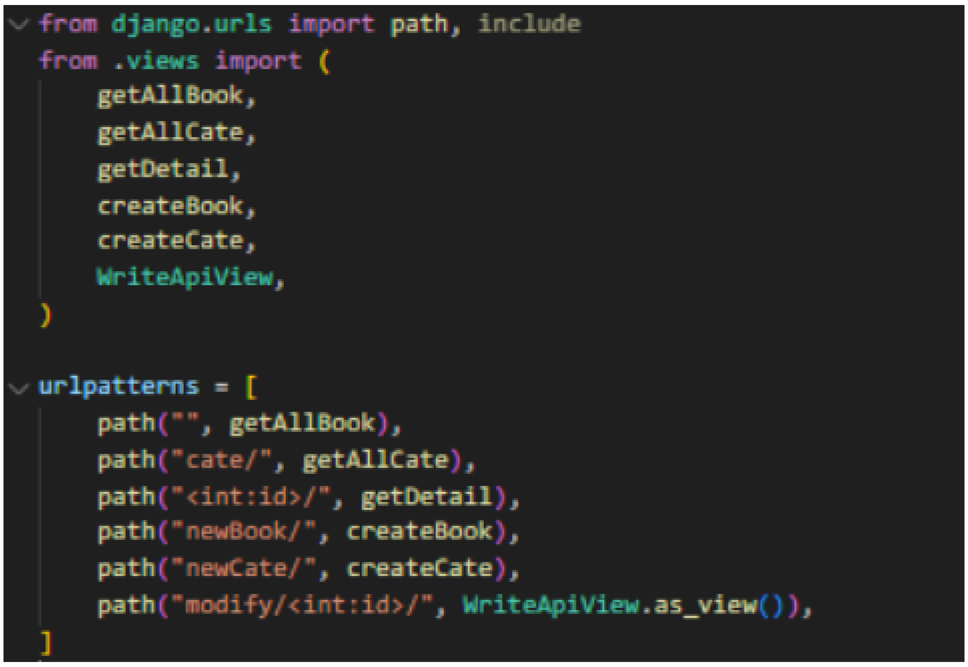


```python
class WriteApiView(APIView):
    # add permission to check if user is authenticated
    permission_classes = [permissions.IsAuthenticated]

    # 1. Update
    def put(self, request, id, *args, **kwargs):
        """
        Updates the todo item with given id if exists
        """
        book = Item.objects.get(id=id)
        if not book:
            return Response(
                {"res": "Object with book id does not exists"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        serializer = ItemSerializer(instance=book, data=request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_200_OK)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    # 2. Delete
    def delete(self, request, id, *args, **kwargs):
        """
        Deletes the todo item with given id if exists
        """
        book = Item.objects.get(id=id)
        if not book:
            return Response(
                {"res": "Object with book id does not exists"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        book.delete()
        return Response({"res": "Object deleted!"}, status=status.HTTP_200_OK)
```



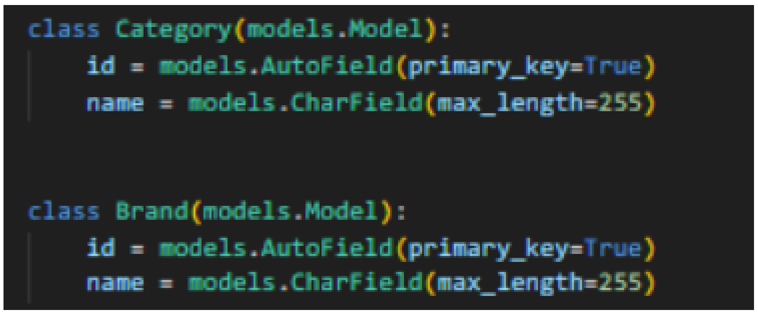
```python
@api_view(["POST"])
@permission_classes([IsAuthenticated])
def createCate(request, *args, **kwargs):
    serializer = CategorySerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(["GET"])
def getAllCate(request, *args, **kwargs):
    cates = Category.objects.all()
    serializer = CategorySerializer(cates, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)
```

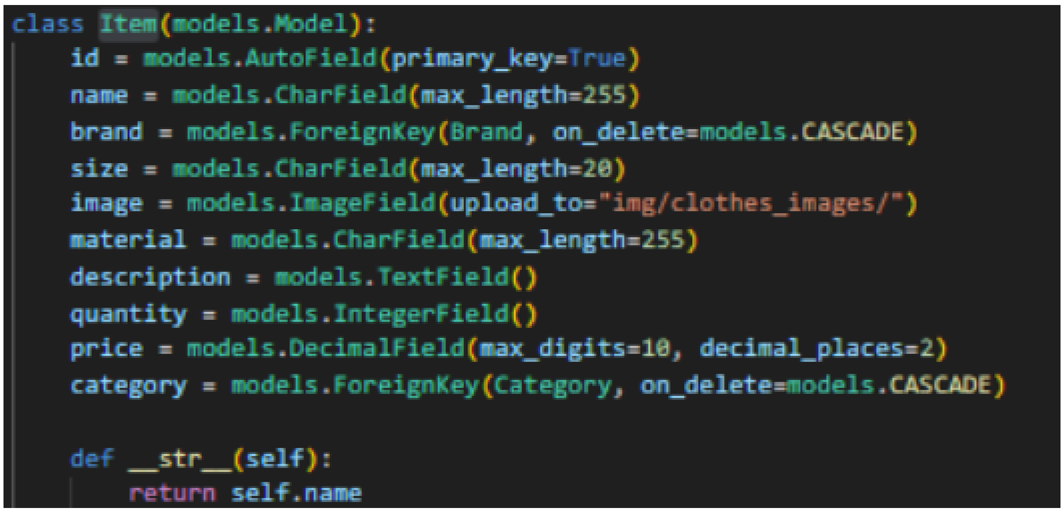
Url:


```python
from django.urls import path, include
from .views import (
    getAllBook,
    getAllCate,
    getDetail,
    createBook,
    createCate,
    WriteApiView,
)

urlpatterns = [
    path("", getAllBook),
    path("cate/", getAllCate),
    path("<int:id>/", getDetail),
    path("newBook/", createBook),
    path("newCate/", createCate),
    path("modify/<int:id>/", WriteApiView.as_view()),
]
```

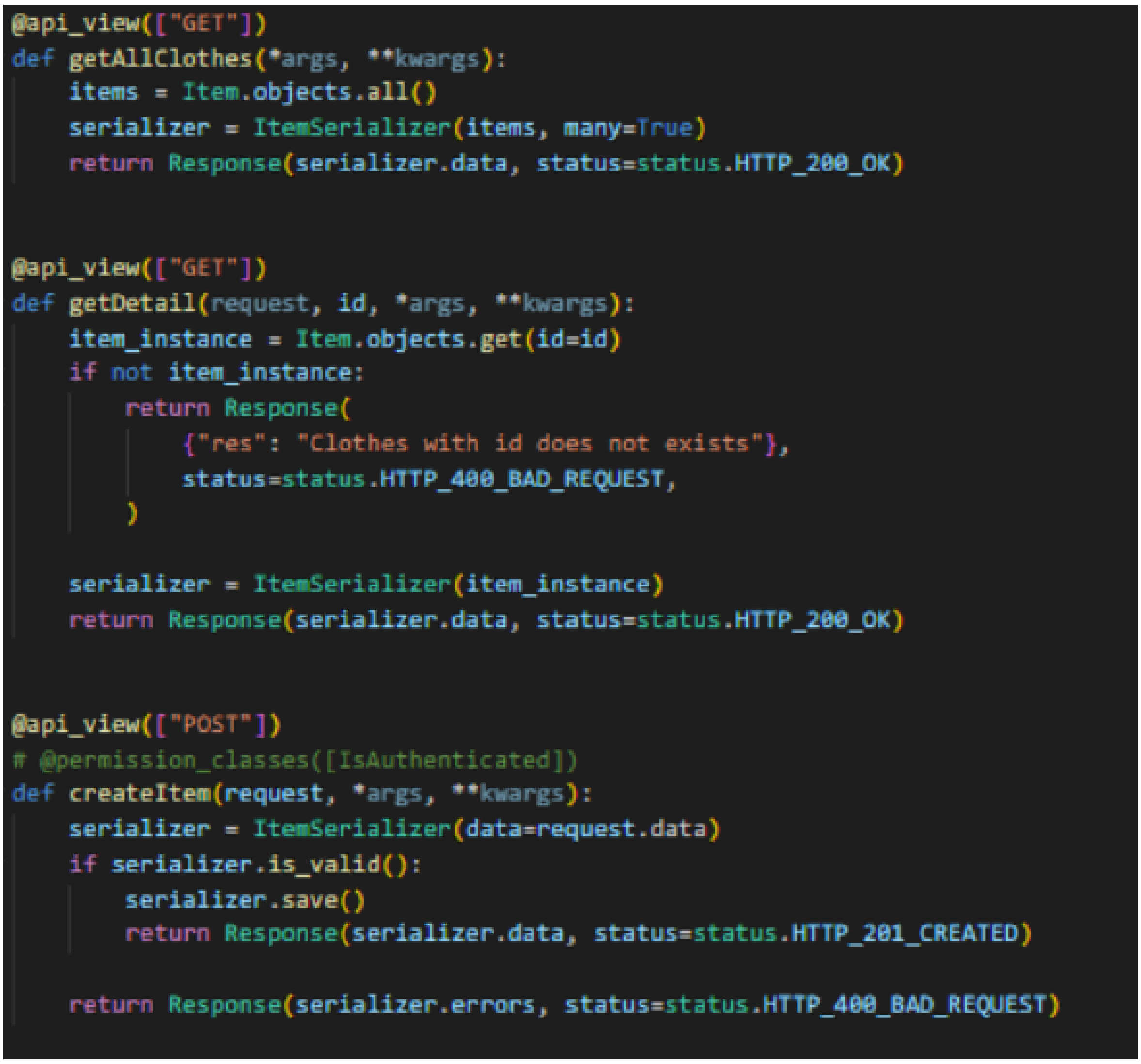
7. App Clothes:

Model:

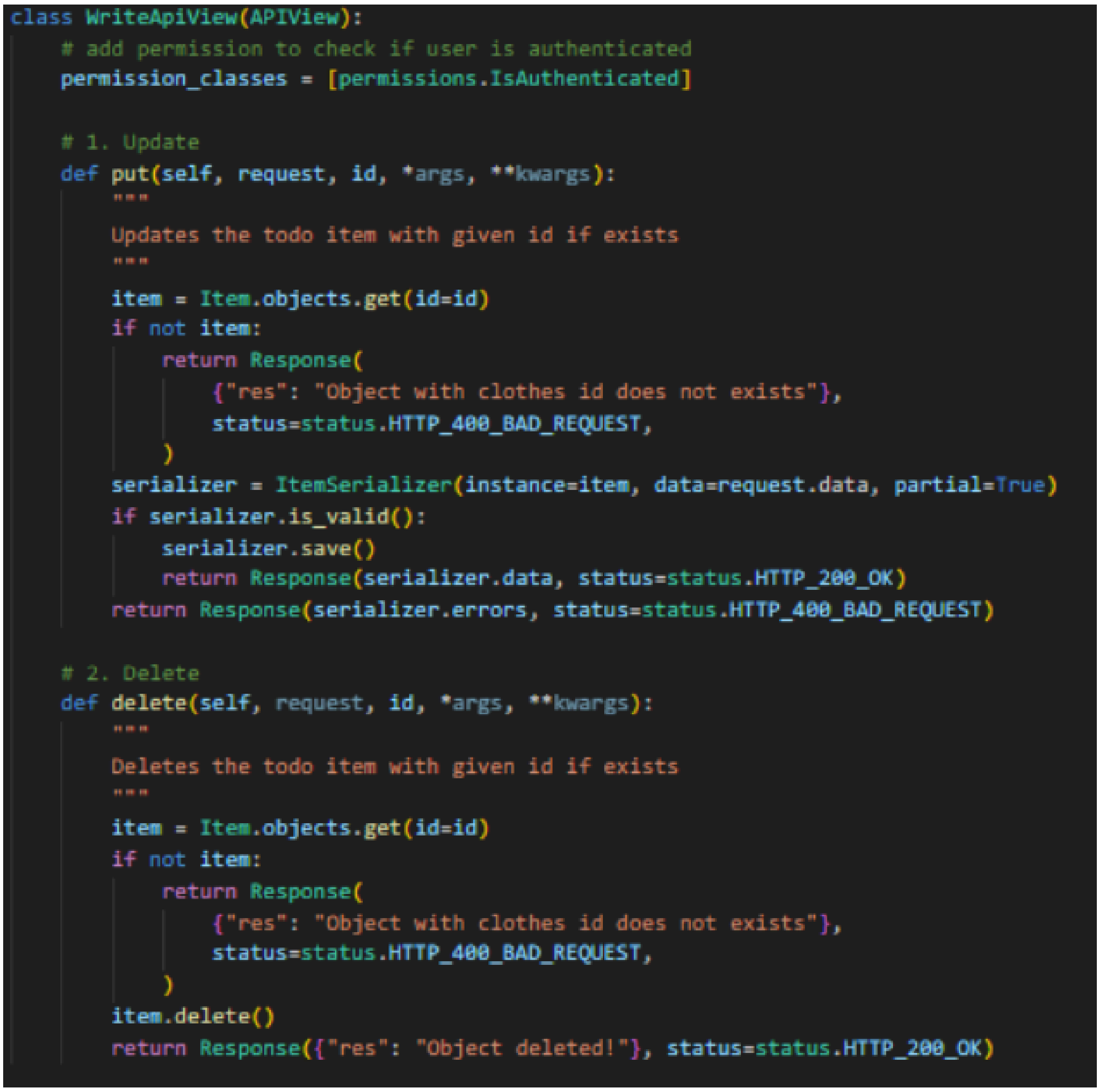

```python
class Category(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=255)


class Brand(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=255)
```


```python
class Item(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=255)
    brand = models.ForeignKey(Brand, on_delete=models.CASCADE)
    size = models.CharField(max_length=20)
    image = models.ImageField(upload_to="img/clothes_images/")
    material = models.CharField(max_length=255)
    description = models.TextField()
    quantity = models.IntegerField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)

    def __str__(self):
        return self.name
```

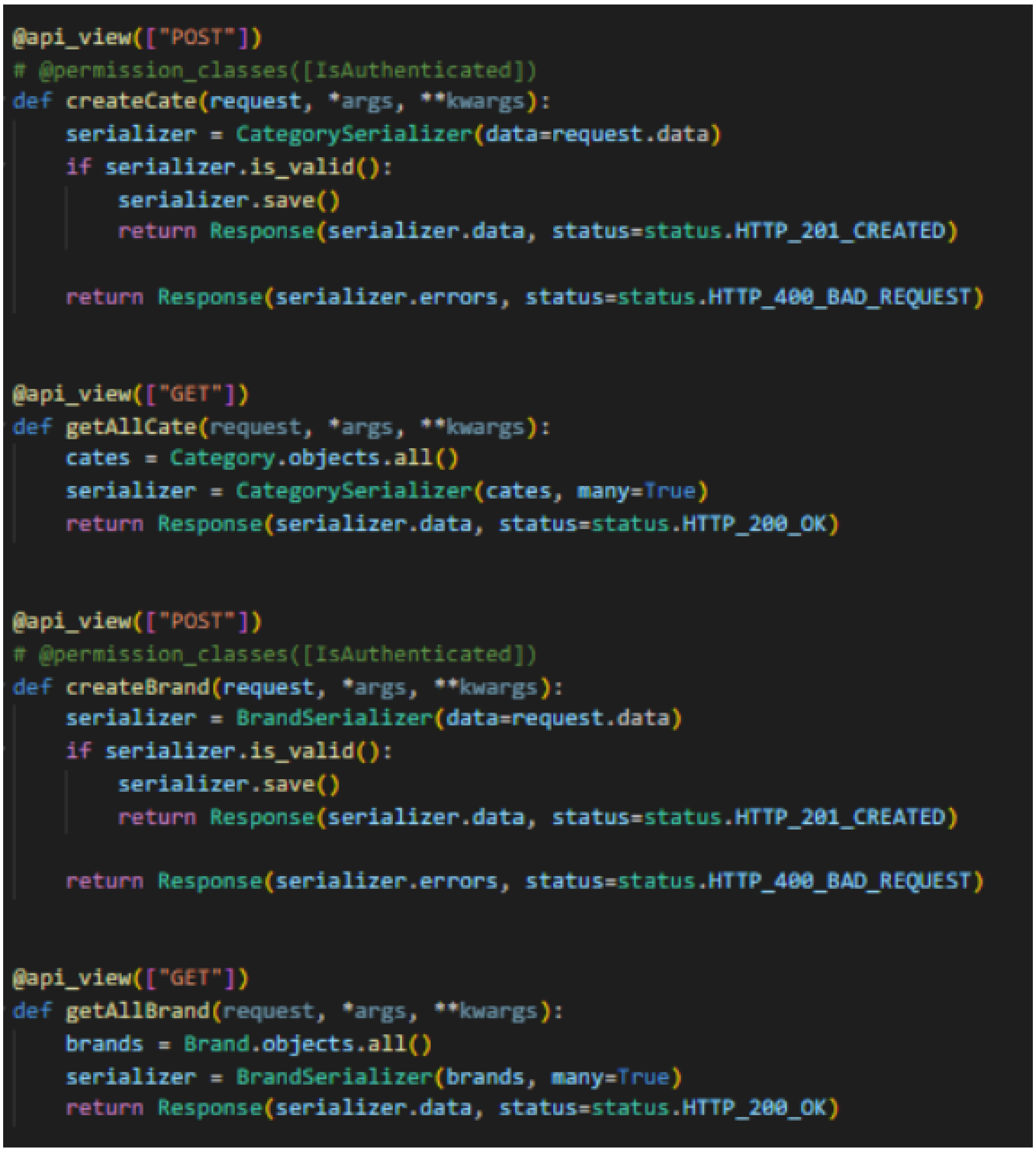
View:



```python
@api_view(["GET"])
def getAllClothes(*args, **kwargs):
    items = Item.objects.all()
    serializer = ItemSerializer(items, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["GET"])
def getDetail(request, id, *args, **kwargs):
    item_instance = Item.objects.get(id=id)
    if not item_instance:
        return Response(
            {"res": "Clothes with id does not exists"},
            status=status.HTTP_400_BAD_REQUEST,
        )

    serializer = ItemSerializer(item_instance)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createItem(request, *args, **kwargs):
    serializer = ItemSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

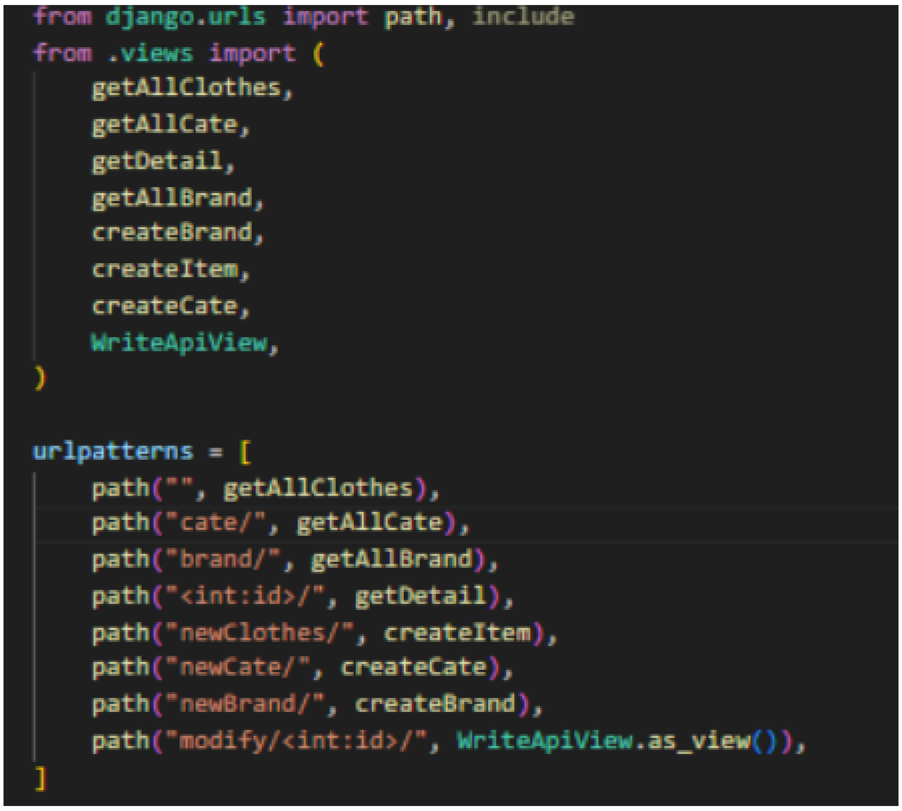


```python
class WriteApiView(APIView):
    # add permission to check if user is authenticated
    permission_classes = [permissions.IsAuthenticated]

    # 1. Update
    def put(self, request, id, *args, **kwargs):
        """
        Updates the todo item with given id if exists
        """
        item = Item.objects.get(id=id)
        if not item:
            return Response(
                {"res": "Object with clothes id does not exists"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        serializer = ItemSerializer(instance=item, data=request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_200_OK)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    # 2. Delete
    def delete(self, request, id, *args, **kwargs):
        """
        Deletes the todo item with given id if exists
        """
        item = Item.objects.get(id=id)
        if not item:
            return Response(
                {"res": "Object with clothes id does not exists"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        item.delete()
        return Response({"res": "Object deleted!"}, status=status.HTTP_200_OK)
```



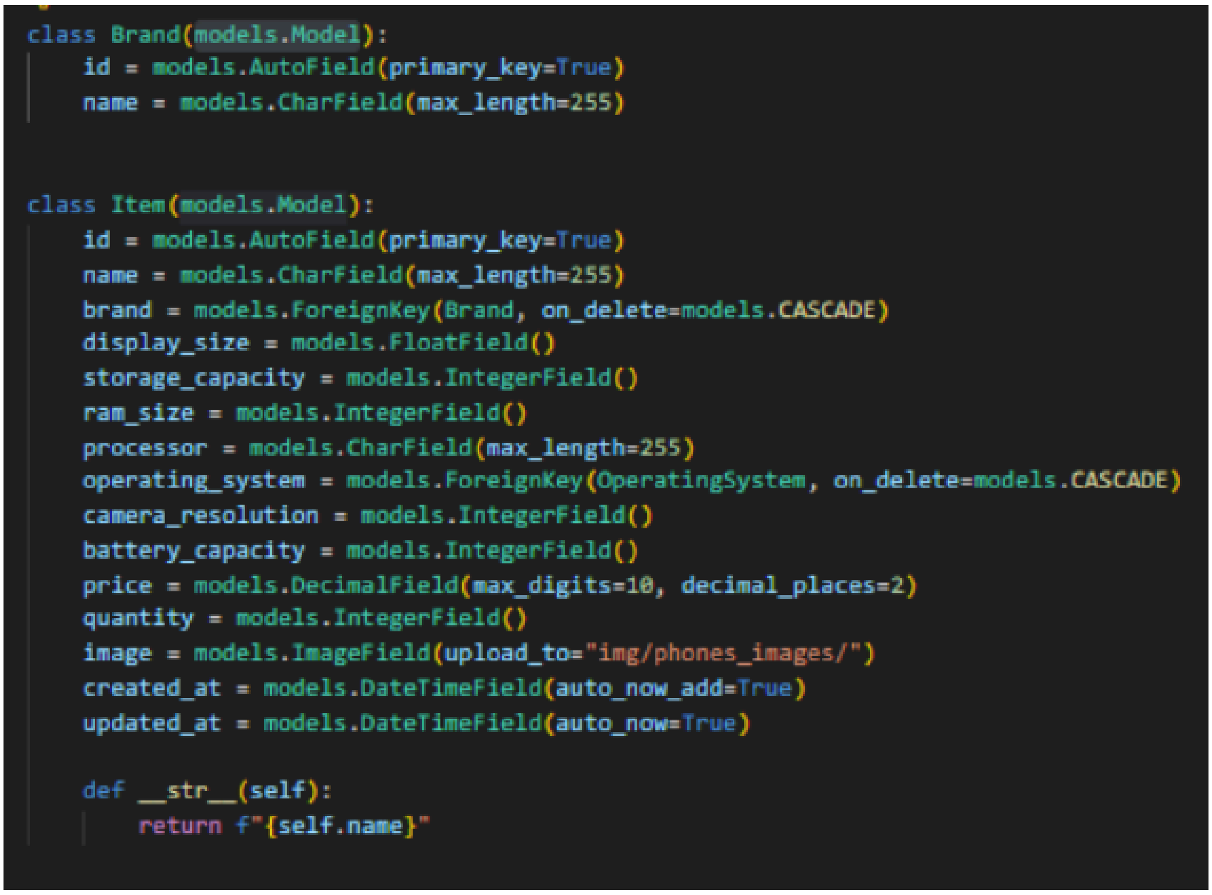
```python
@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createCate(request, *args, **kwargs):
    serializer = CategorySerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(["GET"])
def getAllCate(request, *args, **kwargs):
    cates = Category.objects.all()
    serializer = CategorySerializer(cates, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createBrand(request, *args, **kwargs):
    serializer = BrandSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(["GET"])
def getAllBrand(request, *args, **kwargs):
    brands = Brand.objects.all()
    serializer = BrandSerializer(brands, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)
```

Url:



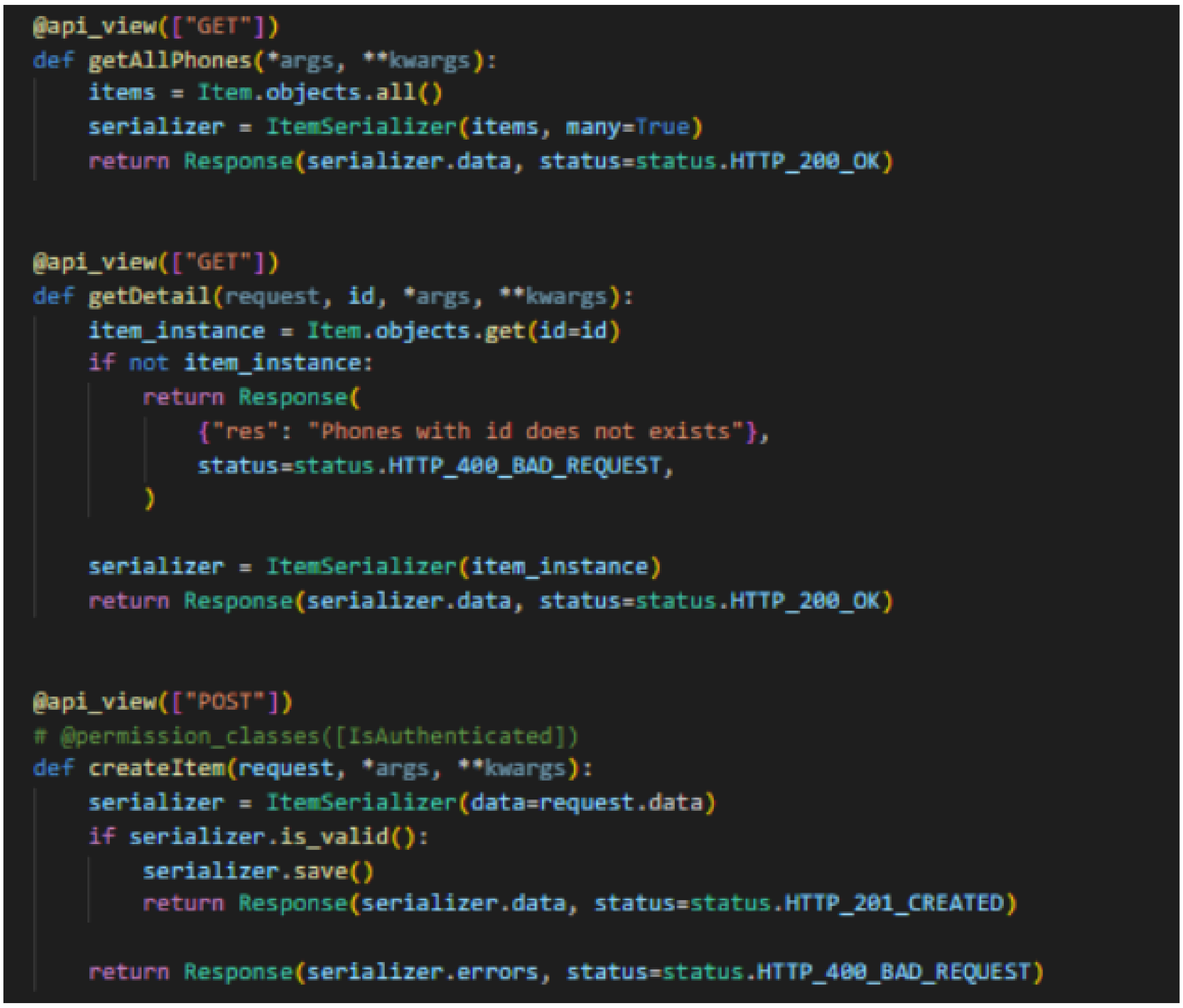
```python
from django.urls import path, include
from .views import (
    getAllClothes,
    getAllCate,
    getDetail,
    getAllBrand,
    createBrand,
    createItem,
    createCate,
    WriteApiView,
)

urlpatterns = [
    path("", getAllClothes),
    path("cate/", getAllCate),
    path("brand/", getAllBrand),
    path("<int:id>/", getDetail),
    path("newClothes/", createItem),
    path("newCate/", createCate),
    path("newBrand/", createBrand),
    path("modify/<int:id>/", WriteApiView.as_view()),
]
```

8. App Phones:
   Model:

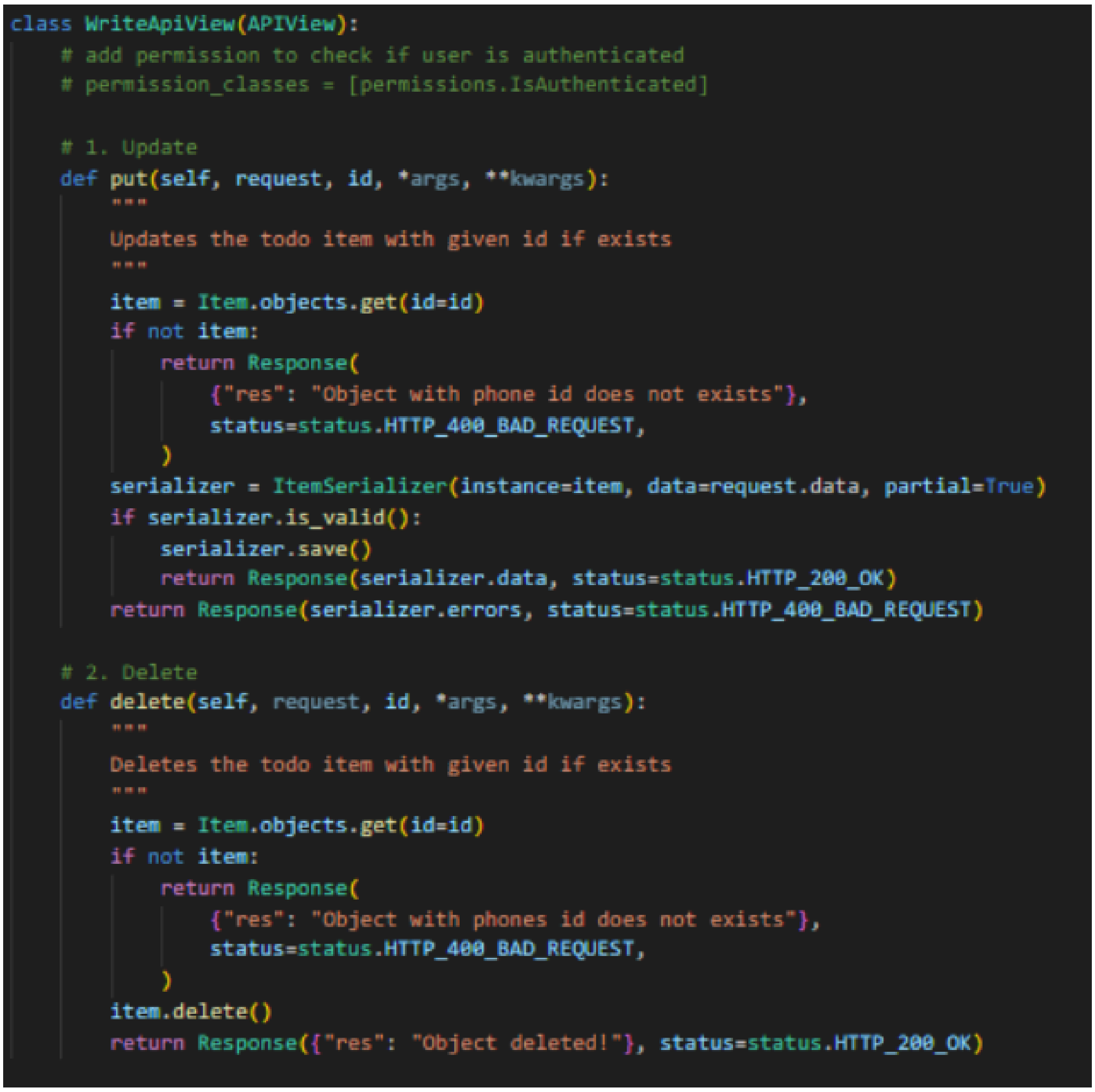


```python
class Brand(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=255)


class Item(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=255)
    brand = models.ForeignKey(Brand, on_delete=models.CASCADE)
    display_size = models.FloatField()
    storage_capacity = models.IntegerField()
    ram_size = models.IntegerField()
    processor = models.CharField(max_length=255)
    operating_system = models.ForeignKey(OperatingSystem, on_delete=models.CASCADE)
    camera_resolution = models.IntegerField()
    battery_capacity = models.IntegerField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    quantity = models.IntegerField()
    image = models.ImageField(upload_to="img/phones_images/")
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.name}"
```

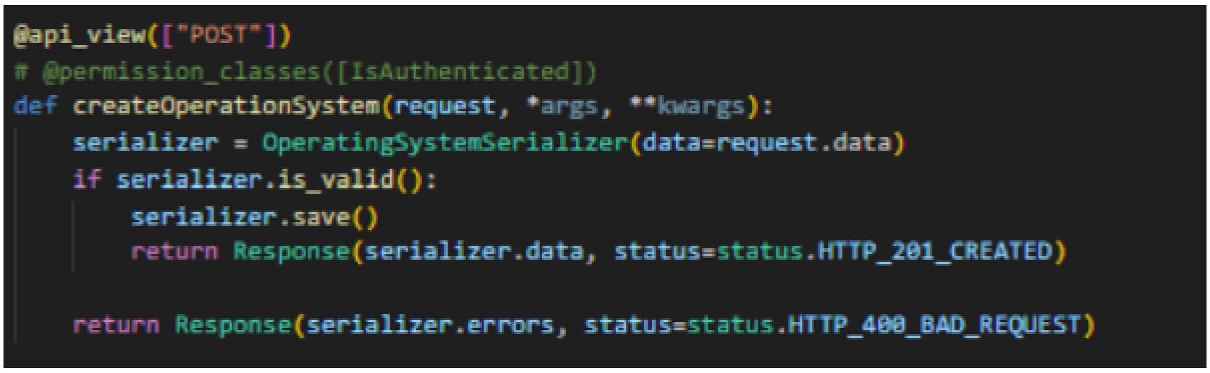
View:



```
@api_view(["GET"])
def getAllPhones(*args, **kwargs):
    items = Item.objects.all()
    serializer = ItemSerializer(items, many=True)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["GET"])
def getDetail(request, id, *args, **kwargs):
    item_instance = Item.objects.get(id=id)
    if not item_instance:
        return Response(
            {"res": "Phones with id does not exists"},
            status=status.HTTP_400_BAD_REQUEST,
        )

    serializer = ItemSerializer(item_instance)
    return Response(serializer.data, status=status.HTTP_200_OK)


@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createItem(request, *args, **kwargs):
    serializer = ItemSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

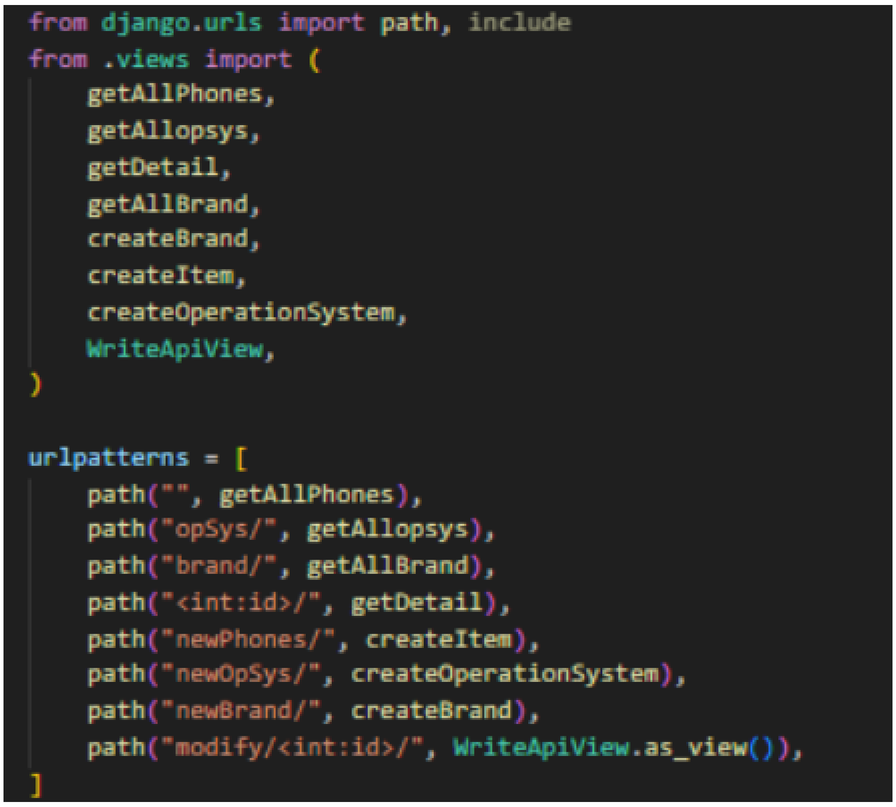


```python
class WriteApiView(APIView):
    # add permission to check if user is authenticated
    # permission_classes = [permissions.IsAuthenticated]

    # 1. Update
    def put(self, request, id, *args, **kwargs):
        """
        Updates the todo item with given id if exists
        """
        item = Item.objects.get(id=id)
        if not item:
            return Response(
                {"res": "Object with phone id does not exists"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        serializer = ItemSerializer(instance=item, data=request.data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_200_OK)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    # 2. Delete
    def delete(self, request, id, *args, **kwargs):
        """
        Deletes the todo item with given id if exists
        """
        item = Item.objects.get(id=id)
        if not item:
            return Response(
                {"res": "Object with phones id does not exists"},
                status=status.HTTP_400_BAD_REQUEST,
            )
        item.delete()
        return Response({"res": "Object deleted!"}, status=status.HTTP_200_OK)
```



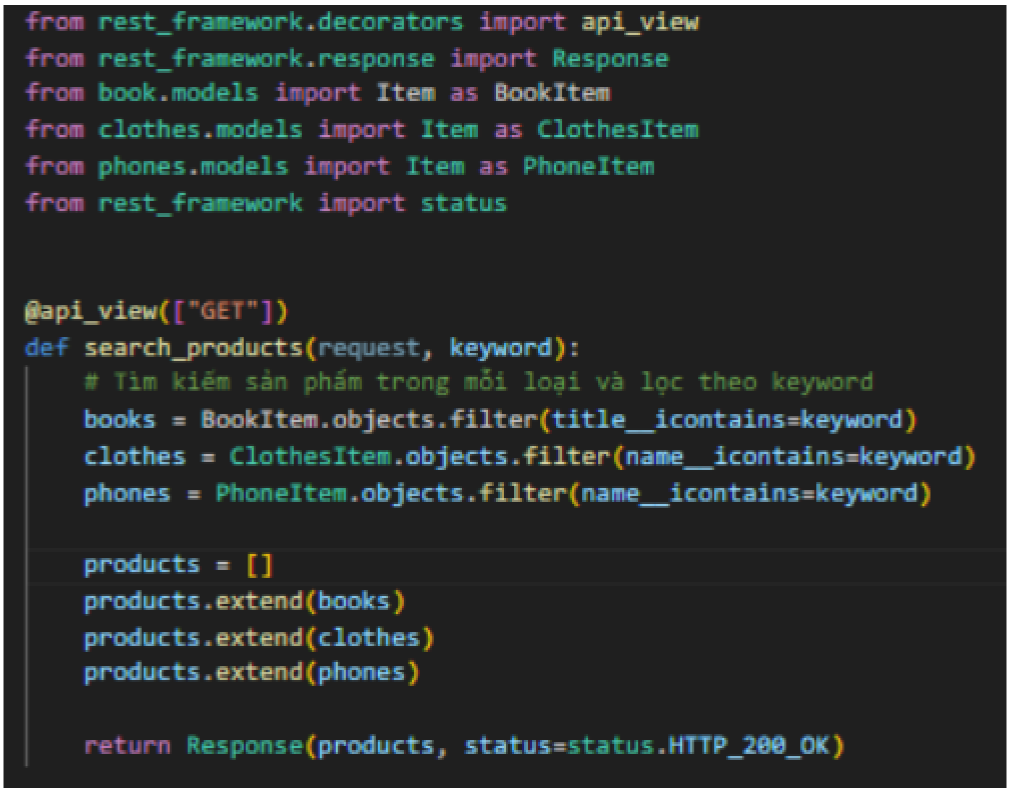
```python
@api_view(["POST"])
# @permission_classes([IsAuthenticated])
def createOperationSystem(request, *args, **kwargs):
    serializer = OperatingSystemSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Url:



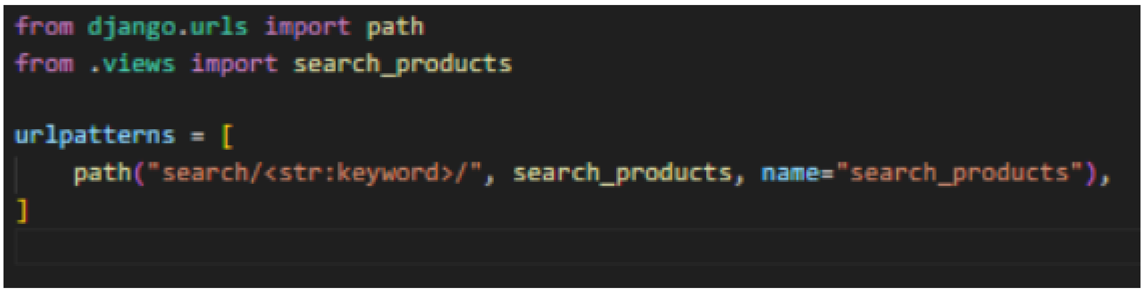
```python
from django.urls import path, include
from .views import (
    getAllPhones,
    getAllopsys,
    getDetail,
    getAllBrand,
    createBrand,
    createItem,
    createOperationSystem,
    WriteApiView,
)

urlpatterns = [
    path("", getAllPhones),
    path("opSys/", getAllopsys),
    path("brand/", getAllBrand),
    path("<int:id>/", getDetail),
    path("newPhones/", createItem),
    path("newOpSys/", createOperationSystem),
    path("newBrand/", createBrand),
    path("modify/<int:id>/", WriteApiView.as_view()),
]
```

9. App Search
   View:



```python
from rest_framework.decorators import api_view
from rest_framework.response import Response
from book.models import Item as BookItem
from clothes.models import Item as ClothesItem
from phones.models import Item as PhoneItem
from rest_framework import status


@api_view(["GET"])
def search_products(request, keyword):
    # Tìm kiếm sản phẩm trong mỗi loại và lọc theo keyword
    books = BookItem.objects.filter(title__icontains=keyword)
    clothes = ClothesItem.objects.filter(name__icontains=keyword)
    phones = PhoneItem.objects.filter(name__icontains=keyword)

    products = []
    products.extend(books)
    products.extend(clothes)
    products.extend(phones)

    return Response(products, status=status.HTTP_200_OK)
```

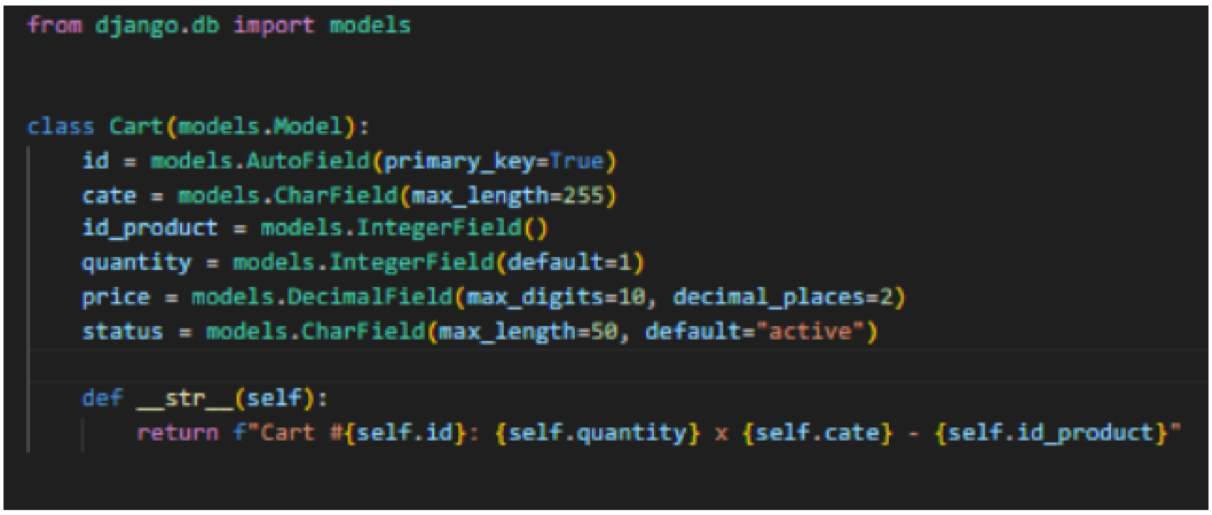
   Url:



```python
from django.urls import path
from .views import search_products

urlpatterns = [
    path("search/<str:keyword>/", search_products, name="search_products"),
]
```

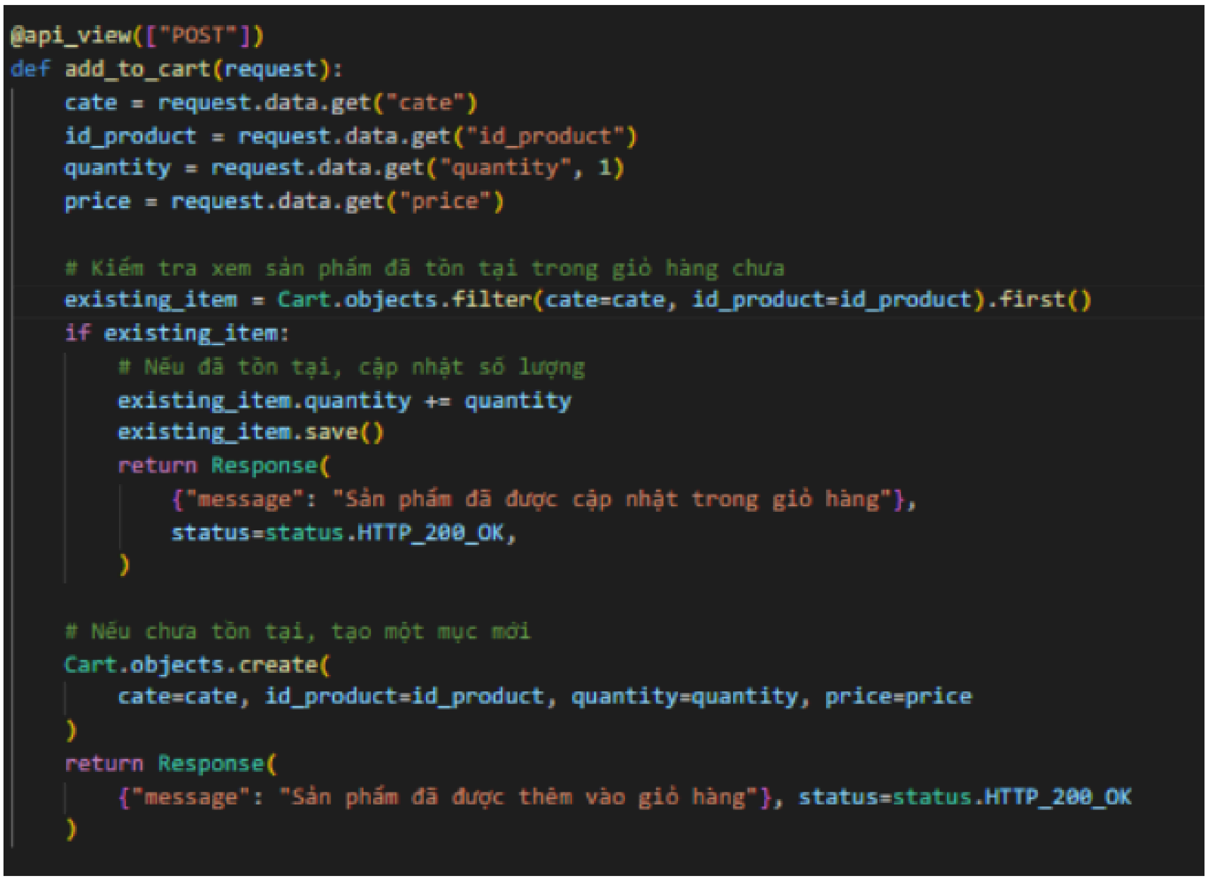
10. App Cart

Model:

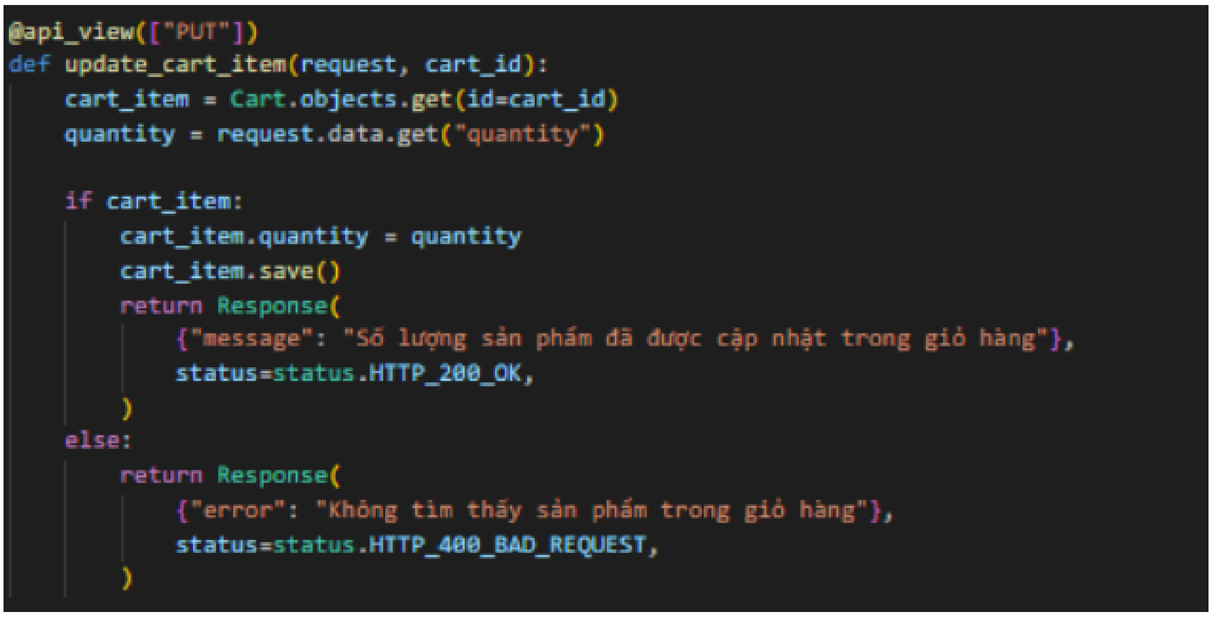


```python
from django.db import models


class Cart(models.Model):
    id = models.AutoField(primary_key=True)
    cate = models.CharField(max_length=255)
    id_product = models.IntegerField()
    quantity = models.IntegerField(default=1)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    status = models.CharField(max_length=50, default="active")

    def __str__(self):
        return f"Cart #{self.id}: {self.quantity} x {self.cate} - {self.id_product}"
```

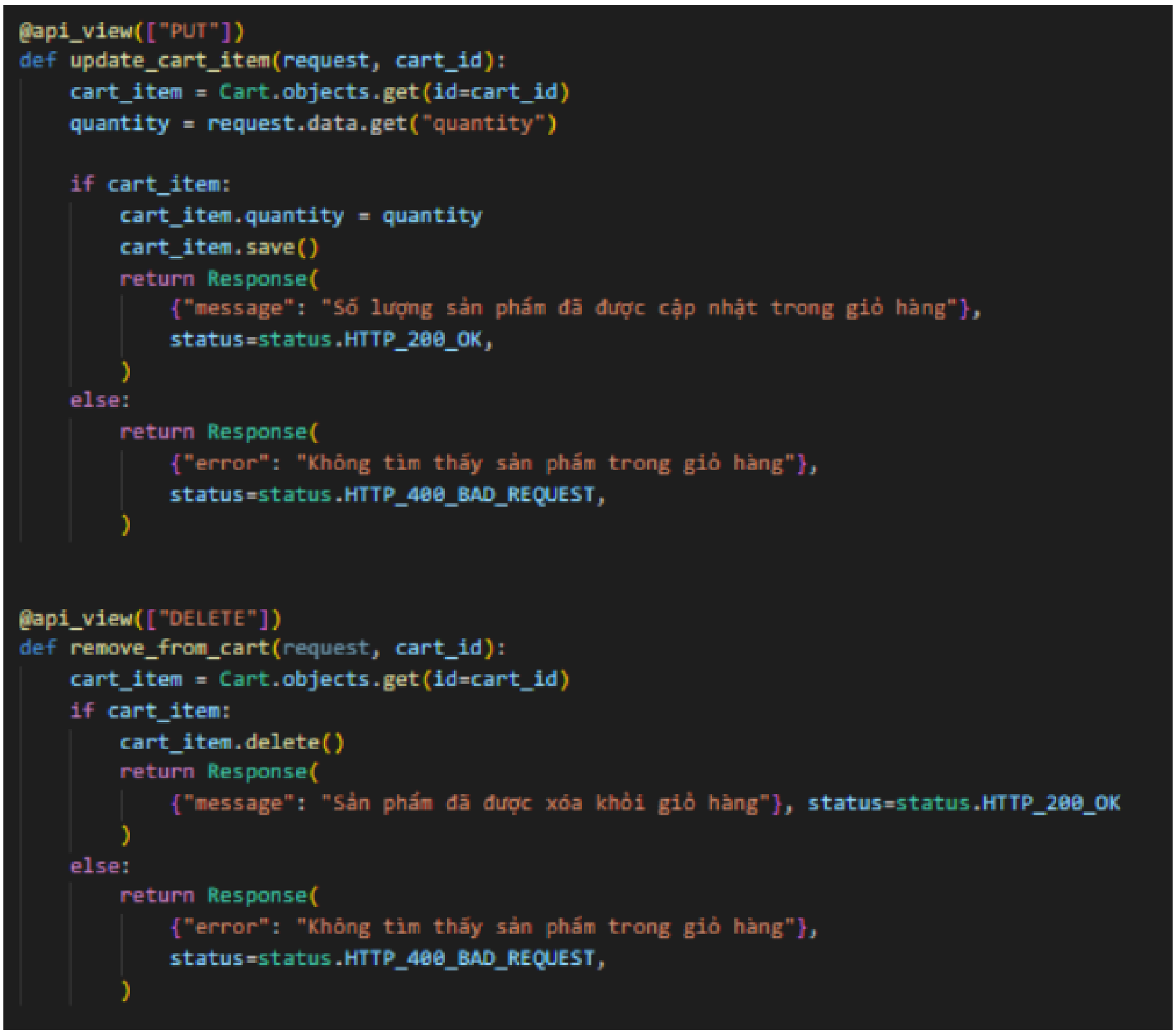
View:



```python
@api_view(["POST"])
def add_to_cart(request):
    cate = request.data.get("cate")
    id_product = request.data.get("id_product")
    quantity = request.data.get("quantity", 1)
    price = request.data.get("price")

    # Kiểm tra xem sản phẩm đã tồn tại trong giỏ hàng chưa
    existing_item = Cart.objects.filter(cate=cate, id_product=id_product).first()
    if existing_item:
        # Nếu đã tồn tại, cập nhật số lượng
        existing_item.quantity += quantity
        existing_item.save()
        return Response(
            {"message": "Sản phẩm đã được cập nhật trong giỏ hàng"},
            status=status.HTTP_200_OK,
        )

    # Nếu chưa tồn tại, tạo một mục mới
    Cart.objects.create(
        cate=cate, id_product=id_product, quantity=quantity, price=price
    )
    return Response(
        {"message": "Sản phẩm đã được thêm vào giỏ hàng"}, status=status.HTTP_200_OK
    )
```

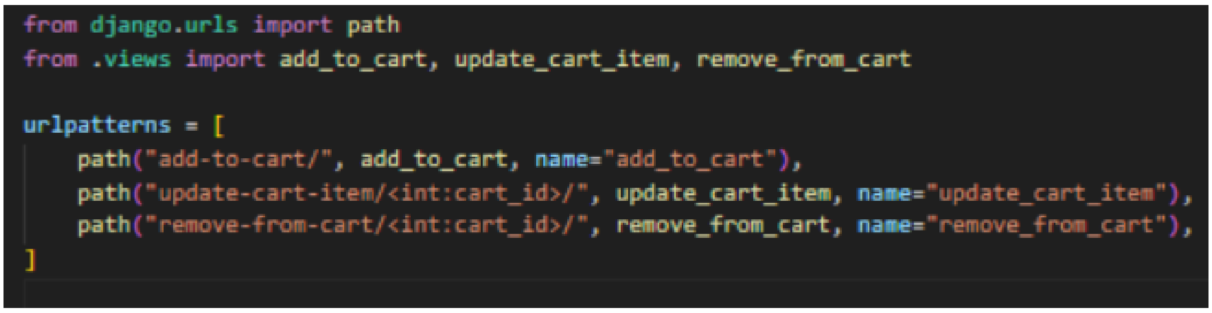


```python
@api_view(["PUT"])
def update_cart_item(request, cart_id):
    cart_item = Cart.objects.get(id=cart_id)
    quantity = request.data.get("quantity")

    if cart_item:
        cart_item.quantity = quantity
        cart_item.save()
        return Response(
            {"message": "Số lượng sản phẩm đã được cập nhật trong giỏ hàng"},
            status=status.HTTP_200_OK,
        )
    else:
        return Response(
            {"error": "Không tìm thấy sản phẩm trong giỏ hàng"},
            status=status.HTTP_400_BAD_REQUEST,
        )
```



```python
@api_view(["PUT"])
def update_cart_item(request, cart_id):
    cart_item = Cart.objects.get(id=cart_id)
    quantity = request.data.get("quantity")

    if cart_item:
        cart_item.quantity = quantity
        cart_item.save()
        return Response(
            {"message": "Số lượng sản phẩm đã được cập nhật trong giỏ hàng"},
            status=status.HTTP_200_OK,
        )
    else:
        return Response(
            {"error": "Không tìm thấy sản phẩm trong giỏ hàng"},
            status=status.HTTP_400_BAD_REQUEST,
        )


@api_view(["DELETE"])
def remove_from_cart(request, cart_id):
    cart_item = Cart.objects.get(id=cart_id)
    if cart_item:
        cart_item.delete()
        return Response(
            {"message": "Sản phẩm đã được xóa khỏi giỏ hàng"}, status=status.HTTP_200_OK
        )
    else:
        return Response(
            {"error": "Không tìm thấy sản phẩm trong giỏ hàng"},
            status=status.HTTP_400_BAD_REQUEST,
        )
```

Url:



```python
from django.urls import path
from .views import add_to_cart, update_cart_item, remove_from_cart

urlpatterns = [
    path("add-to-cart/", add_to_cart, name="add_to_cart"),
    path("update-cart-item/<int:cart_id>/", update_cart_item, name="update_cart_item"),
    path("remove-from-cart/<int:cart_id>/", remove_from_cart, name="remove_from_cart"),
]
```